SQLite

TW . Web Technologies Course
MM, 2020/21 Edition

Sérgio Nunes
DEI, FEUP, U.Porto

Database Management Systems

- ➔ Database Management Systems (DBMS) are software applications that provide features do define, create, maintain, and control databases.
- ➔ DBMS are typically targeted at a specific database model, such as the relational model (RDBMS), the object oriented (OODBMS), or others.
- ➔ DBMS architectures typically follow a client-server model, where clients (e.g. applications) interact with a centralized server managing the database.
- ➔ An alternative architecture is a serverless model where access to the database is done via library calls, without message passing between different processes. The most popular example is SQLite.
- ➔ *Question: Where can we see databases in use?*

DBMS Features

- Features that a DBMS can provide:
  - Data storage, retrieval and update;
  - Constrain enforcement over data;
  - Schema management;
  - Support for transactions and concurrency;
  - Recovery from failure;
  - Access control for data interaction;
  - And many more...
- Popular commercial RDBMS include: Oracle, MS SQL Server, IBM DB2.
- Open-source RDBMS include: PostgreSQL, MySQL, MariaDB, SQLite.

SQLite

- ➔ SQLite is a serverless, ACID compliant, open-source RDBMS.
- ➔ It is commonly used in mobile phones, operating systems, applications (e.g. browsers), amount others. It is the most used database in the world.
- ➔ A SQLite database is a single file.
- ➔ Interaction with a SQLite database is done via library calls.

- ➔ http://sqlite.org

Default (sqlite3)

```
ssn@paco~/Documents/UPorto/FEUP/MIEIC/1920/TW/Sandbox$ ll
total 48
drwxr-xr-x   4 ssn  staff   128B 13 Nov 16:41 .
drwxr-xr-x@ 11 ssn  staff   352B 13 Nov 16:09 ..
-rw-r--r--@  1 ssn  staff   1,8K 13 Nov 16:41 course_projects.sql
-rw-r--r--   1 ssn  staff    20K 13 Nov 16:41 course_projects.sqlite
ssn@paco~/Documents/UPorto/FEUP/MIEIC/1920/TW/Sandbox$ sqlite3 course_projects.sqlite
-- Loading resources from /Users/ssn/.sqliterc
SQLite version 3.28.0 2019-04-15 14:49:49
Enter ".help" for usage hints.
sqlite> .tables
groups              project             student             student_evaluation
sqlite> .schema student
CREATE TABLE student (
  id INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  email TEXT,
  groups_id INTEGER REFERENCES groups
);
sqlite> SELECT * FROM student;
id          name        email       groups_id
----------  ----------  ----------  ----------
1           João        NULL        2
2           Rita        NULL        2
3           Ana         ana@gmail.  3
4           Pedro       NULL        1
5           Michael     NULL        3
6           Rui         NULL        3
7           Maria       NULL        1
8           Susana      NULL        1
sqlite>
```

UNIX command to list directory contents

Open SQLite database

SQLite command to list all existing tables (.tables)

SQLite command to print the schema of a table (.schema students)

SQL command to list all students from student table

DB Browser for SQLite - /Users/ssn/Documents/UPorto/FEUP/MIEIC/1920/TW/Sandbox/course_projects.sqlite

New Database | Open Database | Write Changes | Revert Changes | Open Project | Save Project | Attach Database | Close Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Create Table | Create Index | Modify Table | Delete Table | Print

Name	Type	Schema
Tables (4)		
groups		CREATE TABLE groups (-- Name
id	INTEGER	"id" INTEGER
name	TEXT	"name" TEXT NOT NULL
project		CREATE TABLE project (id INTEG
id	INTEGER	"id" INTEGER
name	TEXT	"name" TEXT NOT NULL
group_id	INTEGER	"group_id" INTEGER
student		CREATE TABLE student (id INTEG
id	INTEGER	"id" INTEGER
name	TEXT	"name" TEXT NOT NULL
email	TEXT	"email" TEXT
groups_id	INTEGER	"groups_id" INTEGER
student_evaluation		CREATE TABLE student_evaluatio
student_id	INTEGER	"student_id" INTEGER
project_id	INTEGER	"project_id" INTEGER
grade	INTEGER	"grade" INTEGER NOT NULL CHE
Indices (0)		
Views (0)		
Triggers (0)		

DB Schema

Name	Type	Schema
Tables (4)		
groups		CREATE TABLE groups (-- Name '
project		CREATE TABLE project (id INTEGE

SQL Log

Show SQL submitted by Application | Clear

```
1 PRAGMA foreign_keys = '1';
2 PRAGMA database_list;
3 SELECT type,name,sql,tbl_name FROM "main".sqlite_master;
4 PRAGMA encoding
5
```

SQL Log | Plot

Edit Database Cell

Mode: Text | Import | Export | Set as NULL

NULL

Type of data currently in cell: NULL

0 byte(s)

Apply

Remote

Identity

Name	Commit	Last modified	Size

UTF-8

Install SQLite

- ➔ SQLite is provided as an executable file.
- ➔ Simply download your operating system version and copy the sqlite3 / sqlite3.exe to your project folder.
- ➔ *If you plan to use SQLite in other projects you should copy SQLite to a folder that is accessible system-wide. This requires some additional knowledge on configuring the system path.*
- ➔ If you open the sqlite3 application (e.g. double click) you will see the SQLite command line interface. You can exit this interface by entering the command ".quit".

Ongoing example

- Download an example SQLite database from:
  - https://drive.google.com/file/d/1Omc8TaqNrvwdlv_tlZTXfHoeaMXlUyFX/view?usp=sharing

- Copy this file to the project's folder, where sqlite3 is.
- Open sqlite3 and then open the database with the command:
  - `.open course_projects.sqlite`

Using SQLite

➔ You can type ".help" in the SQLite command line interface to see a summary of all the available commands.

```
sqlite>
sqlite> .help
.auth ON|OFF            Show authorizer callbacks
.backup ?DB? FILE       Backup DB (default "main") to FILE
.bail on|off            Stop after hitting an error.  Default OFF
.binary on|off          Turn binary output on or off.  Default OFF
.cd DIRECTORY           Change the working directory to DIRECTORY
.changes on|off         Show number of rows changed by SQL
.check GLOB             Fail if output since .testcase does not match
.clone NEWDB            Clone data into NEWDB from the existing database
.databases              List names and files of attached databases
.dbconfig ?op? ?val?    List or change sqlite3_db_config() options
.dbinfo ?DB?            Show status information about the database
.dump ?TABLE? ...       Render all database content as SQL
.echo on|off            Turn command echo on or off
.eqp on|off|full|...    Enable or disable automatic EXPLAIN QUERY PLAN
.excel                  Display the output of next command in a spreadsheet
.exit ?CODE?            Exit this program with return-code CODE
.expert                 EXPERIMENTAL. Suggest indexes for specified queries
.fullschema ?--indent?  Show schema and the content of sqlite_stat tables
.headers on|off         Turn display of headers on or off
.help ?-all? ?PATTERN?  Show help text for PATTERN
.import FILE TABLE      Import data from FILE into TABLE
.imposter INDEX TABLE   Create imposter table TABLE on index INDEX
.indexes ?TABLE?        Show names of indexes
.limit ?LIMIT? ?VAL?    Display or change the value of an SQLITE_LIMIT
.lint OPTIONS           Report potential schema issues.
.log FILE|off           Turn logging on or off.  FILE can be stderr/stdout
.mode MODE ?TABLE?      Set output mode
.nullvalue STRING       Use STRING in place of NULL values
.once (-e|-x|FILE)      Output for the next SQL command only to FILE
.open ?OPTIONS? ?FILE?  Close existing database and reopen FILE
```

```
.nullvalue STRING       Use STRING in place of NULL values
.once (-e|-x|FILE)      Output for the next SQL command only to FILE
.open ?OPTIONS? ?FILE?  Close existing database and reopen FILE
.output ?FILE?          Send output to FILE or stdout if FILE is omitted
.parameter CMD ...      Manage SQL parameter bindings
.print STRING...        Print literal STRING
.progress N             Invoke progress handler after every N opcodes
.prompt MAIN CONTINUE   Replace the standard prompts
.quit                   Exit this program
.read FILE              Read input from FILE
.restore ?DB? FILE      Restore content of DB (default "main") from FILE
.save FILE              Write in-memory database into FILE
.scanstats on|off       Turn sqlite3_stmt_scanstatus() metrics on or off
.schema ?PATTERN?       Show the CREATE statements matching PATTERN
.selftest ?OPTIONS?     Run tests defined in the SELFTEST table
.separator COL ?ROW?    Change the column and row separators
.session ?NAME? CMD ...  Create or control sessions
.sha3sum ...            Compute a SHA3 hash of database content
.shell CMD ARGS...      Run CMD ARGS... in a system shell
.show                   Show the current values for various settings
.stats ?on|off?         Show stats or turn stats on or off
.system CMD ARGS...     Run CMD ARGS... in a system shell
.tables ?TABLE?         List names of tables matching LIKE pattern TABLE
.testcase NAME          Begin redirecting output to 'testcase-out.txt'
.timeout MS             Try opening locked tables for MS milliseconds
.timer on|off           Turn SQL timer on or off
.trace ?OPTIONS?        Output each SQL statement as it is run
.vfsinfo ?AUX?          Information about the top-level VFS
.vfslist                List all available VFSes
.vfsname ?AUX?          Print the name of the VFS stack
.width NUM1 NUM2 ...    Set column widths for "column" mode
sqlite>
```

SQLite Commands

- SQLite commands can be divided in three types:
  - **Meta Commands**, which are used to interact with databases (e.g. open, view, save), view or define SQLite settings, and other administrative operations. These commands start with a dot (.) - .help
  - **SQL Data Definition Language (DDL)**, standard SQL commands to create, alter, and delete database tables.
  - **SQL Data Manipulation Language (DML)**, standard SQL commands to query, add, alter, and delete data.

.help

➔ `.help ?-all? ?PATTERN?`

➔ Show help text for PATTERN.

➔ [.help] shows a list of available commands.

➔ [.help <command>] provides details about a specific command.

```
⌥⌘1                    Default (sqlite3)

sqlite> .help open
.open ?OPTIONS? ?FILE?   Close existing database and reopen FILE
     Options:
        --append        Use appendvfs to append database to the end of FILE
        --deserialize   Load into memory useing sqlite3_deserialize()
        --hexdb         Load the output of "dbtotxt" as an in-memory database
        --maxsize N     Maximum size for --hexdb or --deserialized database
        --new           Initialize FILE to an empty database
        --readonly      Open FILE readonly
        --zip           FILE is a ZIP archive
sqlite>
```

.open

➔ `.open ?OPTIONS? ?FILE?`

➔ Close existing database and reopen FILE.

➔ [.open --new] creates a new empty database.

➔ [.open --readonly] opens FILE in readonly mode.

```
⌥⌘1                    Default (sqlite3)
sqlite>
sqlite>
sqlite>
sqlite> .open course_projects.sqlite
sqlite> .tables
groups              project             student             student_evaluation
sqlite>
```

.tables

➔ `.tables ?TABLE?`

➔ List names of tables matching LIKE pattern TABLE.

➔ [.tables] lists all tables in the currently open database.

.schema

➔ `.schema ?--indent? ?PATTERN?`

➔ Show the CREATE statements matching PATTERN.

➔ The indent option tries to pretty-print the schema.

```
⌥⌘1          Default (sqlite3)

sqlite>
sqlite> .schema --indent student
CREATE TABLE student(
  id INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  email TEXT,
  groups_id INTEGER REFERENCES groups
);
sqlite> .schema --indent project
CREATE TABLE project(
  id INTEGER PRIMARY KEY,
  name TEXT NOT NULL,
  group_id INTEGER REFERENCES groups
);
sqlite>
```

.header

➔ `.headers on|off`

➔ Turn display of headers on or off.

```
sqlite> .header off
sqlite> SELECT * FROM student;
1           João        NULL        2
2           Rita        NULL        2
3           Ana         ana@gmail.  3
4           Pedro       NULL        1
5           Michael     NULL        3
6           Rui         NULL        3
7           Maria       NULL        1
8           Susana      NULL        1
sqlite> .header on
sqlite> SELECT * FROM student;
id          name        email       groups_id
----------  ----------  ----------  ----------
1           João        NULL        2
2           Rita        NULL        2
3           Ana         ana@gmail.  3
4           Pedro       NULL        1
5           Michael     NULL        3
6           Rui         NULL        3
7           Maria       NULL        1
8           Susana      NULL        1
sqlite>
```

.mode

➔ `.mode MODE ?TABLE?`

➔ Set output mode (e.g. csv, column, html, insert, line, etc).

➔ [.mode column] shows results using the default table layout.

```
⌥⌘1          Default (sqlite3)

sqlite> sqlite> .mode csv
sqlite> SELECT * FROM student;
id,name,email,groups_id
1,"João",NULL,2
2,Rita,NULL,2
3,Ana,ana@gmail.com,3
4,Pedro,NULL,1
5,Michael,NULL,3
6,Rui,NULL,3
7,Maria,NULL,1
8,Susana,NULL,1
sqlite>
sqlite>
```

```
⌥⌘1          Default (sqlite3)

sqlite> .mode column
sqlite> SELECT * FROM student;
id          name        email       groups_id
----------  ----------  ----------  ----------
1           João        NULL        2
2           Rita        NULL        2
3           Ana         ana@gmail.  3
4           Pedro       NULL        1
5           Michael     NULL        3
6           Rui         NULL        3
7           Maria       NULL        1
8           Susana      NULL        1
sqlite>
```

.width

➔ `.width NUM1 NUM2 ...`

➔ Set column width (in characters) for column mode.

```
sqlite> SELECT * FROM student;
id          name            email           groups
----------  --------------  --------------  ------
1           João            NULL            2
2           Rita            NULL            2
3           Ana             ana@gmail.com   3
4           Pedro           NULL            1
5           Michael         NULL            3
6           Rui             NULL            3
7           Maria           NULL            1
8           Susana          NULL            1
sqlite>
```

```
sqlite> SELECT * FROM student;
id    name            email           groups
----  --------------  --------------  ------
1     João            NULL            2
2     Rita            NULL            2
3     Ana             ana@gmail.com   3
4     Pedro           NULL            1
5     Michael         NULL            3
6     Rui             NULL            3
7     Maria           NULL            1
8     Susana          NULL            1
sqlite>
```

.read

➔ `.read FILE`

➔ Read input from FILE.

➔ [.read mydb.sql] executes all SQL statements in the mysb.sql file.

.save

➔ `.save FILE`

➔ Write the in-memory database into FILE.

➔ [.save mydb.sqlite] saves the currently active (in-memory) database to the mydb.sqlite file.

Install DB Browser for SQLite

- ➔ DB Browser for SQLite is an open-source graphical user interface (GUI) to interact with SQLite databases.
- ➔ https://sqlitebrowser.org
- ➔ Download and install latest release (menu option "GitHub").
- ➔ DB Browser can be used to open and alter existing databases or to create new SQLite databases.

Data Manipulation Language

➔ Standard SQL can be used to query a database.

➔ `SELECT * FROM students;`

➔
```
SELECT student.name, groups.name
FROM student, groups
WHERE student.groups_id = groups.id;
```

```
⌥⌘1  Default (sqlite3)

sqlite>
sqlite>
sqlite>
sqlite> SELECT * FROM student;
id    name            email           groups
----  --------------  --------------  ------
1     João            NULL            2
2     Rita            NULL            2
3     Ana             ana@gmail.com   3
4     Pedro           NULL            1
5     Michael         NULL            3
6     Rui             NULL            3
7     Maria           NULL            1
8     Susana          NULL            1
sqlite>
```

```
⌥⌘1  Default (sqlite3)

sqlite>
sqlite> SELECT student.name, groups.name
   ...> FROM student, groups
   ...> WHERE student.groups_id = groups.id;
name  name
----  --------------
João  Grupo 2
Rita  Grupo 2
Ana   Grupo 3
Pedr  Grupo 1
Mich  Grupo 3
Rui   Grupo 3
Mari  Grupo 1
Susa  Grupo 1
sqlite>
```

Data Definition Language

➔ Standard SQL can be used to create, alter, or delete tables.

```
➔ CREATE TABLE book (
    id INTEGER PRIMARY KEY,
    name TEXT
  );

➔ ALTER TABLE book RENAME TO books;

➔ ALTER TABLE books
  ADD COLUMN author TEXT;

➔ DROP TABLE books;
```

Creating a Database

- To create a new database you create a text file (script) with all the SQL statements necessary to create the tables.
- Then, you have two ways to create the SQLite database:
  - In SQLite, with the command [.read script_file.sql]
  - In the command line:
    - Linux / OSX: [sqlite3 database.sqlite3 < script_file.sql]
    - Windows: [sqlite3.exe DB.db ".read script_file.sql"]

Populating a Database

- To insert data to a SQLite database, you create a text file with the SQL statements necessary to insert each record to each table.
- You have two ways to execute this SQL script:
  - In SQLite, with the command [.read script_inserts.sql]
  - In the command line:
    - Linux / OSX: [sqlite3 database.sqlite3 < script_inserts.sql]
    - Windows: [sqlite3.exe DB.db ".read script_inserts.sql"]